React Best Practices

Contents

File Organization:

File organization is not only the best practice for react applications, but also it is the best for other applications as well. The file structure of create-react-app is one possible way of organizing your react files. While there is not necessarily one file structure that is better than another, it important to keep your files organized. In React, your file structure will grow rapidly, considering every component has at least one file associated with it. Keep an assets folder that contains top-level CSS, images, and font files. Maintain a helpers folder to put other files for any kind of file for functionalities. Keep all the files related to a component into one folder. Usually, the component folder contains more than a single component file, such as test file CSS and one or more component files. If there are any minor components used by a particular component only, better to keep those all small components in the component folder.

When you use Redux in your React project you can use Rails-style pattern for files. In Rails-style pattern, separate folders are used for "actions", "constants", "reducers", "containers", and "components".

A) Tiny and functional components

As we all know, React will work with large components. But if we break them into small sizes, we can reuse them. Small components are easier to read, test, maintain, and reuse. Most beginners in React create Class components even when they aren't using component state or life cycle methods. Functional components are more efficient for simple components.

Advantages of using Functional Components.

- Less code
- Easier to understand
- Stateless
- Easier to test
- There is no this binding.
- Easier to extract smaller components.

When you are using functional component, you have no control over the re-rendering process. When something changes or even component changes itself, React will re-render functional components. In former react versions has a solution to use React.PureComponent. PureComponent allows shallow props and state comparison. When props or content of the component or component itself changed component will re-render. Otherwise, PureComponent skip re-render and reuse the last rendered result instead.

Reusability components:

Each functional component should have one function that means one functional component is equal to one function. When you are creating a functional component with one function you can improve the reusability of that component.

Delete Redundant code

Not only in React but also in all application development the common rule is keeping the code is succinct and tiny as much as possible. React best practices instruct to keep the error-free code and incisive code. Don't Repeat Yourself (DRY) is a principle of software development focused at minimizing repetition of software patterns, replacing it with abstractions or using data normalization to avoid redundancy. In code fomatting you can use your own style guide or use a popular fully-fledged style guide (Airbnb React/JSX Style Guide, Facebook Style Guide, etc). If you start to follow anyone of the style follow that don't confuse with others.

https://github.com/airbnb/javascript/tree/master/react

Unnecessary <div>s

When Creating React component it is important to keep in mind that you are still building an HTML document. People tend to get divitis in React which ultimately leads to incorrect HTML. We can use another approach that uses <React.Fragment> tags. <React.Fragment> was introduced in React v16.2, we can use them instead of the extraneous <div> tag.

Necessary Comments only

Add a comment in the application where necessary. Removing the ability to comment from an application meant I HAVE to write literate code, no exceptions. It gives untidy free code sections. In general, comments are a wart that stipulates poor design, especially long-winded comments where its clear the developer didn't have a clue what the heck they are doing and tried to make up for it by writing a comment.

Understand to handle 'this'

Since functional components don't require this binding, you'll want to use them whenever possible. But if you are using ES6 class, you'll want to bind this manually since React doesn't auto bind the function within that component. You can bind in props or in the function itself.

Naming in the end

Name a function or a component after typing the scripts because they should be easily identifiable. For example, you choose the name of a component like FacebookButton instantly because of the component code. But in the future, you may use that component as TwitterButton,YoutubeButton. So, the best practice is to name that component as Button. Normally when you finish the function, you should be able to choose the common name for the component and function. Naming in the end increases the reusability.

Use Upper Camel Case Names

When you are working in React remember that you are using JSX (JavaScript Extension) instead of HTML. The component created by you should be named in the upper camel case, a.k.a Pascal Case. The upper camel case means words are written without spaces, and the first letter of each word is capitalized. For example, If there is a component named as selectbutton then you should name it as SelectButton instead of selectbutton. Using the upper camel case helps to JSX to differentiate the default JSX element tags from created elements. However, you can use lower case letters to name a component but it is not a best practice.

Use Prettier and snippet libraries

Prettier is a code formatting tool. Prettier has a set of rules for code formatting and indention. You can use Sonarlint to check spells, function length and suggestions for better approaches. Using Husky is not only a good practice for React but also a good practice for Git. You can define the husky in package.json file. Husky prevents your application from bad commit and bad push.

Code snippets help you to code best and trend syntax. They keep your code relatively error-free. You can use a lot of snippet libraries such as ES7 React, JavaScript (ES6) code snippets, etc.